Amendments to the Specification

Please replace paragraph **[0024]** with the following rewritten paragraph:

[0024] In the second stage of the weaving process, the combined, or woven, program code block 130 is compiled into one or more executable modules 140. As shown in Fig. 1, in this second stage, the aspect-oriented weaver 110 takes one of the woven code blocks 130. The aspect-oriented weaver 110 compiles the woven variables ~~131~~ 133 and the woven processes ~~133~~ 134 of that woven code block 130 into one or more executable modules 140.

Please replace paragraph **[0042]** with the following rewritten paragraph:

[0042] As shown in Fig. 2, in various exemplary embodiments, the first stage is a syntax stage 210. In this syntax stage 210, no local information from a given high-level code block 122 has been discarded and no contextual information is available. This corresponds to the original syntax of that high-level code block 122. In the systems, methods and aspect-oriented programming environments according to this invention, the aspect-oriented weaver 110 "weaves" the syntax stage 210 from the high-level code block 122. The syntax stage 210 includes a woven code block 220. The syntax stage ~~220~~ 210 then outputs the woven code block 220 generated in the syntax stage 210 to a next intermediary stage 230. The next intermediate stage 230 is woven by the aspect-oriented weaver 110 from the woven code block 220. After each stage 210, 230, 250 and 270, one or more propagators, that have been defined in a corresponding woven code block 220, 240, or 260 relative to that stage 210, 230 or 250, are run to determine the projections defined in that stage 210, 230 or 250. These projections are then available during the following stages 230, 250 and 270, respectively.

Please replace paragraph **[0045]** with the following rewritten paragraph:

[0045] As shown in Fig. 2, at the last intermediate stage 250, all optimization weaving is completed. The woven code block 260 output by this last intermediate stage 250 is input to the final weaving stage 270. It should be appreciated that the final weaving stage

270 is often the processed value stage. Then, the woven code ~~280~~ output by from the final weaves stage 270 is compiled into an executable program module or set of executable program modules.

Please replace paragraph **[0089]** with the following rewritten paragraph:

[0089] Again, the aspect-oriented weaver 110 invokes a projector 350 by sending an instruction along an inter-propagator visibility path 345. The projector 350 examines the second-stage woven code block 340 for significances that may be effected by future weaving. For example, in the exemplary embodiment shown in ~~Fig. 3~~Fig. 4, the projector 350 determines that the significance E fits this criteria. The projector 350 creates a propagator 355, using a projector/propagator path 347, for the significance E. Then, the propagator 355 communicates with the propagator 335 using an inter-propagator visibility path 337, to determine if anything needs to be updated. In this particular example, there are no significances in common. That is, there is no significance that is continued in both of the first and second stage woven code blocks 320 and 340. Therefore, no updating is needed.

Please replace paragraph **[0095]** with the following rewritten paragraph:

[0095] Subsequently, the aspect-oriented weaver 110 provides instructions 475 to the computation stage 470 to reduce the loop structures contained in the second stage woven code block 460. In response, the computation stage 470 outputs the third stage woven code block 480, which is the final, fully woven stage. This final stage code block 480 is also called the value stage code block 480. The final stage code block 480 is then output to a compiler 490. The compiler 490, under control of the aspect-oriented weaver 110 over the signal channel 495, compiles the fully woven value stage block 480 to form an executable code block ~~500~~. The executable code block ~~500~~ is output from the compiler 490.